Semantic Federation of Distributed Neurodata

Alban Gaignard[1], Johan Montagnat[1], Catherine Faron Zucker[1], and Olivier Corby[2]

[1] I3S laboratory, CNRS & University of Nice Sophia Antipolis, France
http://www.i3s.unice.fr
[2] INRIA Sophia Antipolis, France
http://www-sop.inria.fr

Abstract. Neurodata repositories federation is increasingly needed to implement multi-centric studies. Data federation is difficult due to the heterogeneous nature of distributed data repositories and the technical difficulties faced in addressing simultaneously multiple data sources. This paper describes a data federation system based on a semantic mediation layer and an advanced distributed query engine able to interface to multiple heterogeneous neurodata sources. Both performance and usability indicators are shown, demonstrating the soundness of the approach and its practical feasibility.

1 Introduction

Computational neurosciences are highly dependent on the availability of large neurological datasets. An increasing effort has been invested in distributed computing infrastructures to face the challenges raised by modern multi-centric neuroscience studies [1,2,3]. In particular, distributed neurodata federation is considered to harness the large neurodata resources available [4,5,6]. Federation of heterogeneous data sources is a complex problem due to the need to align the different data models used, and the lack of tools to efficiently query and process distributed databases. The Semantic Web community has developed methodologies and tools to clearly define the semantics of data using models and computer-interpretable metadata. This paper describes how these approaches can be applied to neurodata alignment, and introduce new methods to integrate legacy neurodata sources in a federated platform.

An approach often considered to deliver federated data repositories is *data warehousing* which consists in statically importing all data sources in a centralized repository after semantic alignment. Although simple, this solution has many drawbacks such as the need to transform legacy data entities to different representations, the scalability of the centralized repository, the central point of failure thus created, etc. When considering medical data, it is often even not possible to create a central data repository for legal and ethical reasons. The alternative solution studied in this paper is a *dynamic distributed data federation* mechanism which aligns the data models and the query capability of the heterogeneous repositories to deliver a unified view over the complete data set.

This approach pushes queries to the distributed data sources and gathers results. From the client point of view, distributed data sources are thus virtually integrated as if a single data source was queried. It raises new issues such as the need for expressing queries that can be mapped to multiple data sources, network communication overhead, coherency of distributed queries, and transforming data entities on-the-fly to different representations while preserving their semantics.

We developed the KGRAM distributed query system to align multiple neurodata sources semantically and map queries to the heterogeneous federation. System utilisability is considered by taking into account both the query language expressiveness and performance issues. KGRAM is generic in the sense that it does not require any prior knowledge on the data sources content, and robust in a distributed environment as it transparently adapts to topology changes caused by the addition or the removal of data sources.

2 Distributed data query engine

Different neurodata repositories generally use different data models and store heterogeneous data. The dynamic data federation approach addresses simultaneously the problems of data source heterogeneity and data distribution. It relies on a central federator, and a set of federated data providers. From a unique query, the federator is responsible for the coherent sub-querying of the federated data providers and for unifying all results found into a global result set. It avoids consistency and synchronization issues generally observed in data warehouses where data transformation is performed periodically.

A single main query language is needed to express federation-wide queries. The SPARQL[3] language v1.1 is considered in this work as a highly expressive, versatile, and *de facto* standard semantic data query language. Semantic data is viewed as a collection of triple patterns formalized as (s, p, o), where a subject s is linked to an object o through a predicate p, and usually represented as RDF triples[4]. A collection of triples constitutes a knowledge graph. Semantic querying through SPARQL is equivalent to the matching of a graph-based query pattern into the complete knowledge graph.

More precisely, the SPARQL-based query engine KGRAM described below (i) transforms a graph-based semantic query into the target data sources query languages and (ii) combines the triple results to assemble the reply at runtime. To efficiently perform distributed queries, the engine implements several optimizations including *query rewriting* and *intermediate results exploitation*.

2.1 The KGRAM query engine

KGRAM stands for *Knowledge Graph Abstract Machine* [7]. It is a versatile system aiming at representing, querying and reasoning over semantic data represented as Knowledge Graphs [8]. It enables querying different data source

[3] SPARQL: `http://www.w3.org/TR/rdf-sparql-query/`
[4] RDF: `http://www.w3.org/TR/rdf-primer/`

models, provided that they are able to produce a graph view of their data, *e.g.* RDF but also XML or relational databases. For the sake of generality, KGRAM introduces a set of abstract operators manipulating abstract graph data structures (abstract *Nodes* and *Edges* forming abstract *Graphs*). *Graphs* are navigated through so-called *Producers* responsible for the iteration over their *Nodes* and *Edges*. For each data source kind, a specific *Producer* is needed that abstracts its representation to answer SPARQL queries by returning data represented as graph elements. Knowledge graphs are matched with queries through *Matchers*. Depending on their implementations, the comparison may consist in simple labels equality or it may take into account domain knowledge through class and property subsumption (RDS entailment) or eventually approximate matching based on semantic similarities. Finally, *Filter* operators are responsible for evaluating value constraints over knowledge graphs.

2.2 Distributed query processing

In KGRAM, the evaluation of SPARQL queries basically consists in searching for matching edges in a knowledge graph delivered by a *Producer*. To mashup linked data over distributed data sources, KGRAM introduces the notion of *Metaproducer* responsible for the enumeration of *Nodes* and *Edges* coming from several *Producers* interfaced to several data sources. All producers implement the same query interface, receiving the same messages from the Metaproducer and rewriting the input queries into native data source ones, to shield KGRAM core from data source heterogeneity. To handle distributed data repositories, we extended KGRAM implementation by providing a web service implementation of its *Producer* interface. Figure 1 illustrates the main elements involved in the semantic distributed query processing.

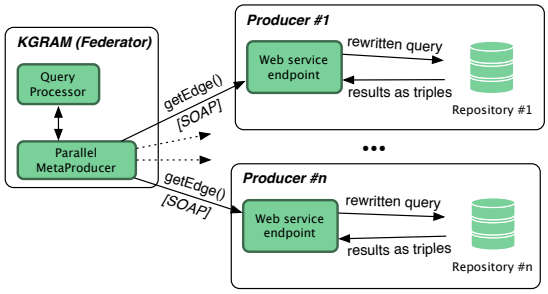

Fig. 1. KGRAM distributed query architecture

Web service endpoints are queried over the standard SOAP protocol. Each remote *Producer* interfaces with its database. It is responsible for enumerating on-the-fly triples matching the SPARQL queries received. Various *Producer* implementations enable querying various data models. The *ParallelMetaProducer*

is responsible for the exploitation of service parallelism and for merging triples coming from the distributed data sources into the resulting knowledge graph. Algorithm 1 illustrates how the *ParallelMetaProducer* distributes a SPARQL query over a set of *Producers*.

Algorithm 1: Naive parallel distributed query processing, with an explicit wait condition.

Data: *Producers* the set of SPARQL endpoints,
TriplePatterns the set of triple patterns forming a basic graph pattern in the initial SPARQL query,
scheduler a thread pool allowing for parallel execution.
Result: *Results* the set of SPARQL results.

```
1 foreach (tp ∈ TriplePatterns) do
2     foreach (p ∈ Producers) do in parallel
3         scheduler.submit(p.matchingTriples(tp)) ;
4     while (not scheduler.isFinished()) do
          ;                                        // synchronization barrier
5     foreach (task ∈ scheduler.getFinished()) do
6         Results ← scheduler.getTask().getResults() ;
```

The principle of the algorithm consists in iterating over each triple pattern forming the initial SPARQL query (line 1). For each triple pattern, all remote *Producers* are queried concurrently (line 3). The federator then waits for all federated endpoints to finish through a synchronization barrier (line 4). Results are finally accumulated for the current triple pattern (lines 5 and 6) and the next triple pattern iteration can be processed (line 1). To soften the synchronization barrier a pipelining strategy was implemented, in which a synchronized blocking queue allows the federator to post-process results as soon as they become available. Due to space constraints, the pipelining algorithms are not detailed.

2.3 Distributed query optimization strategies

In KGRAM, the query distribution principle consists in iterating over each triple pattern request occurring in the SPARQL query. Each triple pattern request is dynamically wrapped into a unitary SPARQL query and pushed to remote data sources. Resulting triples are returned back and accumulated into the KGRAM result graph. To enhance the efficiency of distributed query processing, a set of static and dynamic optimizations were implemented.

The *filter pushing* strategy consists in analyzing the initial SPARQL query to extract value constraints expressed in FILTER clauses. When iterating over each triple pattern request, each applicable value constraint is extracted from the initial FILTER and propagated as a new FILTER clause added to the triple

pattern query. This reduces drastically the communication cost since it prevents the federator from transferring irrelevant results that would be finally filtered.

The same idea is exploited at query runtime through *bind joins*. The KGRAM query processor manages an index of already known mappings between variables and values, thus forming intermediate results. This cache is exploited at query runtime in order to dynamically replace triple pattern variables by their associated values in the queries pushed to federated endpoints. Similarly to the *filter pushing* strategy, *bind joins* reduce drastically the size of transferred results.

2.4 Relational data sources

In neurosciences, it is common that the underlying legacy environments rely on traditional relational databases. KGRAM comes with a default implementation of its *Producer* interface for RDF sources. To cope with relational sources, a specific *Producer* that rewrites triple patterns forming the initial SPARQL query into SQL queries is needed. Query results can then be mapped to the variables of the original SPARQL query to build result graph triples. A generic SQL *Producer* is out of scope of this paper. An ad-hoc *Producer* implementation was considered in the case of the NeuroLOG platform described below.

3 Experimentation in the NeuroLOG platform

The NeuroLOG platform [9] federates data and computational resources from collaborating neuroscience centers. The prime objective of NeuroLOG is to adapt non-invasively to the legacy environments deployed in each participating center, so that each site remains autonomous in the management of its internal resources and tools, while benefiting from the multi-centric medical studies support from the middleware. The platform is federating 5 neuroscience centers spread over France. On each site, the data source is composed of raw neuroradiology files, and description metadata linked to these files (information on image data acquisition, data content, neuropsychological tests associated to images, etc) [6]. The source metadata is often represented and managed in relational databases for historical reasons.

Blue components in Figure 2 sketches the architecture of the NeuroLOG middleware. On each neuroscience site an independently managed legacy relational database is deployed. It is completed by a NeuroLOG middleware database. The multi-centric studies conducted by neuroscientists may be perceived under several facets, involving both the native relational data representation and a semantic data representation enabling richer queries. The *DataFederator* commercial tool[5] is used to dynamically federate relational data sources into a unified view. It can perform SQL queries that are distributed over all platform data sources. It includes both a mediation layer that aligns heterogeneous relational databases

[5] Data Federator: `http://www.sap.com/solutions/sapbusinessobjects/large/information-management/data-integration`

schemas, and rewrite SQL queries applying to the federated view to match the various source schemas. The data mediation semantic alignment is based on a domain ontology, called OntoNeuroLOG that was developed in the context of this project. A federated relational schema is derived from OntoNeuroLOG, serving as the federated view schema. In addition to this relational representation, a semantic representation of the same data sources was created to enable richer querying features delivered by Semantic Web query engines. A centralized approach was adopted, where all relational data sources are mapped to RDF triples (using the MetaMORPHOSES tool[6]) and aggregated in a unique semantic repository. The NeuroLOG platform thus exposes a dual view of the federation metadata, enabling both dynamic SQL querying and static SPARQL querying.

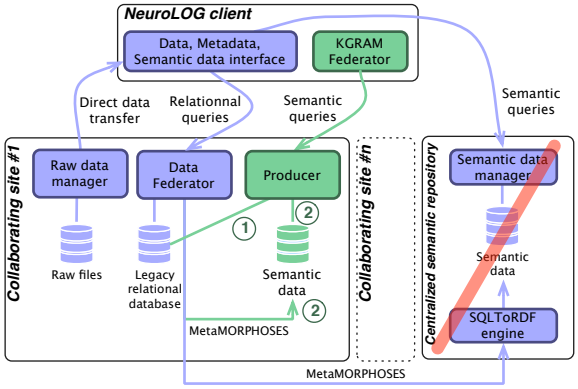

Fig. 2. Data management layer of the NeuroLOG platform

Although very flexible, this system is also confusing for end users due to the dual view of all data entities. The semantic repository is also subject to limitations of a static, centralized warehouse. To overcome these issues, the NeuroLOG platform was extended with the KGRAM query engine introduced in section 2.1 (see green components in Figure 2). A KGRAM remote producer was deployed on top of each site legacy database. This endpoint exposes the site data content in RDF through its *Producer*. Depending on the site deployment option, it either interfaces dynamically to the site native relational database (option ① in Figure 2), or accesses an RDF repository representation of this legacy database (option ②) which is periodically regenerated using MetaMORPHOSES. Consequently, NeuroLOG's centralized RDF repository (crossed over in red in Figure 2) is not needed anymore. It is replaced by a completely dynamic semantic federation (all sites using ①), a distributed version periodically updated (all sites using option ②), or a mixture of both. This setting enables the unified querying

[6] MetaMORPHOSES: `http://metamorphoses.sourceforge.net`

of the RDF repositories and the platform legacy relational databases through the SPARQL language. It solves the central repository limitation, distributing the query load over the federation data servers. It proposes a single view over all data. The experiments reported below demonstrate the query expressiveness and the performance of the KGRAM engine. The NeuroLOG platform both provides a real use case for these experiments and serves as a performance measurement reference.

3.1 Experimental setup

Three query environments are compared. The *relational federation* only exposes heterogeneous relational databases, virtually integrated through the *DataFederator* commercial middleware. It corresponds to the seminal NeuroLOG platform deployment. Two other environments (*semantic federations*) expose heterogeneous data sources virtually integrated through the KGRAM framework. In the *RDF semantic federation* environment, all sites are configured with option ②. It corresponds to a modification of the NeuroLOG platform where the central semantic repository is spread over all participating sites but native repositories are transformed to RDF before being accessed. In the *RDF+SQL semantic federation* environment, a mixture of RDF sources and SQL sources (dynamically accessed through ①) is used to evaluate a completely heterogeneous set up.

Figure 3 (top) illustrates a real clinical use case involving querying neurodata distributed over several centers. It aims at searching for datasets (acquired with Gadolinium contrast agent) associated to patients (join performed at line 4) in the context of multi-centric studies addressing the Multiple Sclerosis pathology. During the evaluation of this query, all types of medical images will be searched in order to match a "GADO" tag (indicating use of the Gadolinium contrast agent). But the evaluation of this query can be considerably enhanced by exploiting clinical knowledge represented in an RDFS vocabulary. Indeed Gadolinium is only used in the context of the MR modality. By exploiting this domain knowledge with KGRAM, the query time can be significantly reduced since all non-MR datasets are excluded. This results from the triple pattern (`?dataset rdf:type dataset:MRDataset`) added to the query which leads, under RDFS entailment, to less intermediate results to be transferred and thus an overall faster distributed query evaluation.

The bottom query from Figure 3 achieves the same clinical objective but makes use of SPARQL property path expressions. This language feature aims at representing paths between two resources by only specifying, in the form of patterns, the sequence of mandatory, optional, reverse, or multiple repetition of properties linking the resources together. It brings a high expressivity to SPARQL queries and it is particularly adapted in the context of graph-based querying. This kind of query cannot be easily expressed in SQL. It is thus difficult to implement it with traditional relational databases. As a full SPARQL 1.1 interpreter, compliant with property path expressions, KGRAM allows for performing this kind of graph-based information retrieval against SQL data sources, which would not have been possible wit traditional SQL query engines.

```
SELECT distinct ?patient ?study ?dataset ?dsName WHERE {
        ?dataset linguistic-expression:has-for-name ?dsName .
        ?patient examination-subject:has-for-subject-identifier ?clinID .
        ?patient iec:is-referred-to-by ?dataset .
        ?study study:involves-as-patient ?patient .
FILTER (regex(?clinID, 'MS') && regex(?dsName, 'GADO')) }

SELECT distinct ?patient ?study ?dsName WHERE {
        ?patient iec:is-referred-to-by/linguistic-exp:has-for-name ?dsName .
        ?patient examination-subject:has-for-subject-identifier ?clinID .
        ?study study:involves-as-patient ?patient .
FILTER (regex(?clinID, 'MS') && regex(?dsName, 'GADO')) }
```

Fig. 3. Top: sample SPARQL query *Q1*, aiming at retrieving patient, study and dataset information in the context of the Multiple Sclerosis disease. Bottom: SPARQL 1.1 property path expressions simplifying the previous query.

3.2 Qualitative evaluation

KGRAM implements the RDFS entailment regime (subsumption between classes or properties), and other inferences based on algebraic properties like the transitivity or symmetry of properties and inverse properties. It thus provides a richer query interface to legacy databases participating into the federation. In addition, in the context of collaborative platforms exposing legacy relational databases, we argue that the design of queries is more intuitive through knowledge-based languages such as SPARQL than through traditional relational languages such as SQL. Indeed, the navigation through links between entities is explicit in SPARQL while it is implicit in SQL and generally requires for intermediate joins. The following SQL query corresponds to the SPARQL query of Figure 3:

```
SELECT Subject.subject_id, Subject.subject_common_identifier, Dataset.name
FROM Study, Subject, Dataset, Rel_Subject_Study AND
WHERE Rel_Subject_Study.Subject_subject_id = Subject.subject_id AND
  Rel_Subject_Study.Study_study_id = Study.study_id AND
  Dataset.Subject_subject_id = Subject.subject_id AND
  Subject.subject_common_identifier LIKE '%MS%' AND Dataset.name LIKE '%GADO%'
```

Whereas joins are naturally expressed in the SPARQL query (line 4 of Figure 3), it is not the case in SQL since a join table may be needed (`Rel_Subject_Study` table, line 3) and must be explicit (line 4, 5 and 6). This definitely complicates the query design, generally considered as complex, error-prone, and time consuming.

3.3 Quantitative evaluation

The distributed query processing times of the *relational federation* (using *DataFederator*) and both *semantic federations* (RDF and RDF+SQL) are compared using KGRAM through two queries. Query *Q1* corresponds to Figure 3 and query

Q2 searches for datasets acquired through the T2-weighted MRI modality. *Q2* leads to only 5 results and is thus, a very selective query. To be robust against variability observed in real distributed systems, results are averaged over three query runs. The average query execution time ± one standard deviation is displayed in the following table showing that for *Q1*, leading to 336 remote invocations, the query times are lower with the optimized SQL federation engine *DataFederator* than with the semantic federation, but it remains in the same order of magnitude. For very selective queries such as *Q2*, we observe comparable query times for all environments:

	Semantic federation: RDF	*or* RDF+SQL	*Relational federation*
Q1 (s)	6.13 ± 0.05	11.76 ± 0.05	3.03 ± 0.25
Q2 (s)	0.60 ± 0.03	1.53 ± 0.14	1.52 ± 0.62

4 Concluding remarks

Transparent data federation approaches generally address performance issues. For instance, DARQ [10], SPLENDID [11] or FedX [12] propose a set of static and dynamic optimizations. KGRAM implements similar strategies to exploit value constraints at query rewriting time and to enhance the querying at runtime through bind joins. Moreover, in line with FedX, KGRAM also exploits the parallelism of both distributed data sources and modern multi-core CPUs with a multithreaded implementation. There remain other optimization opportunities for KGRAM such as data sources selection and subsequent query planning techniques, but to the best of our knowledge, none of the state-of-the art transparent federating approaches address both the issues of efficient and heterogeneous distributed querying. The strength of the KGRAM model is to provide an extensible framework allowing to efficiently and dynamically access to multiple query sources internally using different data representations. In the future, the query rewriting capability of KGRAM is also thought to be a mean to implement fine-grained access control to data sources.

KGRAM eases the federation of distributed and heterogeneous neurodata repositories. Concrete examples in the context of the NeuroLOG platform, mixing both semantic and relational repositories, demonstrate the feasibility and the efficiency of this approach. Beyond performance, the query environment is enriched by the high expressivity of semantic query language. End-users gain from using knowledge with a well-defined semantics. Moreover, through an ontology and its associated entailments, the querying process can exploit domain knowledge to implement smart queries. The current implementation of KGRAM enables read-only access to data sources. Modifying heterogeneous distributed data sources is a challenging problem that we intend to tackle in the future.

Acknowledgments

This work was partly funded by the French National Agency for Research under grants ANR-06-TLOG-024 (NeuroLOG project).

References

1. Geddes, J., Mackay, C., Lloyd, S., Simpson, A., Power, D., Russell, D., Katzarova, M., Rossor, M., Fox, N., Fletcher, J., Hill, D., McLeish, K., Hajnal, J., Lawrie, S., Job, D., McIntosh, A., Wardlaw, J., Sandercock, P., Palmer, J., Perry, D., Procter, R., Ure, J., Bath, P., Watson, G.: The Challenges of Developing a Collaborative Data and Compute Grid for Neurosciences. In: 19th IEEE International Symposium on Computer-Based Medical Systems, Salt Lake City, Utah, IEEE Computer Society (June 2006) 81–86
2. Helmer, K.G., Ambite, J.L., Ames, J., Ananthakrishnan, R., Burns, G., Chervenak, A.L., Foster, I., Liming, L., Keator, D., Macciardi, F., Madduri, R., Navarro, J.P., Potkin, S., Rosen, B., Ruffins, S., Schuler, R., Jessica, A.T., Toga, A., Williams, C., Kesselman, C.: Enabling collaborative research using the Biomedical Informatics Research Network (BIRN). Journal American Medical Informatics Association 2011;18:416-422 doi:10.1136/amiajnl-2010-000032 **18**(4) (April 2011) 416–422
3. Anjum, A., Bloodsworth, P., Habib, I., Lansdale, T., McClatchey, R., Mehmood, Y.: Reusable Services from the neuGRID Project for Grid-Based Health Applications. In: HealthGrid'09, IOS Press (June 2009) 283–288
4. Barillot, C., Benali, H., Dojat, M., Gaignard, A., Gibaud, B., Kinkingnéhun, S., Matsumoto, J.P., Pélégrini-Issac, M., Simon, E., Temal, L.: Federating Distributed and Heterogeneous Information Sources in Neuroimaging: The NeuroBase Project. In: HealthGrid'06, Spain (June 2006) 3–13
5. Weiner, M.W., Veitch, D.P., Aisen, P.S., Beckett, L.A., Cairns, N.J., Green, R.C., Harvey, D., Jack, C.R., Jagust, W., Liu, E., Morris, J.C., Petersen, R.C., Saykin, A.J., Schmidt, M.E., Shaw, L., Siuciak, J.A., Soares, H., Toga, A.W., Trojanowski, J.Q.: The Alzheimer?s Disease Neuroimaging Initiative: A review of papers published since its inception. Alzheimer's and Dementia **8**(1, Supplement) (February 2012) S1–S68
6. Michel, F., Gaignard, A., Ahmad, F., Barillot, C., Batrancourt, B., Dojat, M., Gibaud, B., Girard, P., Godard, D., Kassel, G., Lingrand, D., Malandain, G., Montagnat, J., Pélégrini-Issac, M., Pennec, X., Rojas Balderrama, J., Wali, B.: Grid-wide neuroimaging data federation in the context of the NeuroLOG project. In: HealthGrid'10, Paris, France, IOS Press (June 2010) 112–123
7. Corby, O., Zucker, C.F.: The kgram abstract machine for knowledge graph querying. Web Intelligence and Intelligent Agent Technology, IEEE/WIC/ACM International Conference on **1** (2010) 338–341
8. Corby, O., Dieng-Kuntz, R., Faron-Zucker, C.: Querying the semantic web with corese search engine. In de Mántaras, R.L., Saitta, L., eds.: ECAI, IOS Press (2004) 705–709
9. Montagnat, J., Gaignard, A., Lingrand, D., Rojas Balderrama, J., Collet, P., Lahire, P.: NeuroLOG: a community-driven middleware design. In: HealthGrid. , Chicago, IOS Press (June 2008) 49–58
10. Quilitz, B., Leser, U.: Querying distributed rdf data sources with sparql. The Semantic Web Research and Applications **5021** (2008) 524–538
11. Görlitz, O., Staab, S.: SPLENDID: SPARQL Endpoint Federation Exploiting VOID Descriptions. In: Proceedings of the 2nd International Workshop on Consuming Linked Data, Bonn, Germany (2011)
12. Schwarte, A., Haase, P., Hose, K., Schenkel, R., Schmidt, M.: Fedx: optimization techniques for federated query processing on linked data. In: Proceedings of the 10th international conference on The semantic web - Volume Part I. ISWC'11, Berlin, Heidelberg, Springer-Verlag (2011) 601–616